Don Morgan

The Control Loop

In this series on motion control, we have already discussed permanent magnet brushed and brushless motors, PWM, and several control techniques. This month, we will examine some of the details behind the controller algorithms involved in proportional-integral-derivative (PID) designs. I'll introduce and define some terminology, as well as provide some generally accepted tuning techniques for these algorithms.

Many controllers are available on the market. You may even be designing your own. The descriptions provided here will apply generally, but specific implementations of the algorithms vary, as does terminology.

The control loop

In Figure 1, we have a simple block diagram of a system featuring a control and a machine. The *drive*, or power converter, is the device used to apply power to a system. We discussed such a system in last month's column. ("Hijacked!" October 2000, p. 149.)

Disturbance is a term we'll use to identify the interactive and reactive elements involved in the machine. This may include changes in load, temperature, friction, or whatever might affect the performance of the machine.

The *plant* is responsible for the system response; it is usually passive and dissipates power. In our case it is some kind of motor connected to a load. A motor integrates the drive from the power converter and is thus something like a low pass filter with inherent phase lag.

Feedback can come from different types of sensors. It's specific to the activity of the machine and could be anything from an accelerometer to a laser. In the system we'll be looking at, feedback comes from an encoder.

The heart of the controller lies in the implementation of control laws. If you say the output of a black box is equal to one-half the input, this is that box's control law.

Tuning a PID controller is pretty straightforward, once you know how to start and what steps to follow.

The standard equation for PID control is:

$$U[t] = K_d \frac{de}{dt} + K_p e[t] + K_i \int_{x=0}^{x=t} e[x]dx$$

In this equation, e is the error. Error is derived from the feedback. K_p, K_i, and K_d each represent a constant coefficient. The subscripts represent that with which the coefficient is associated. A pseudocode snippet for this algorithm might look like this:

```
error = cmd - actl;
sum = oldsum + error;
U = Kp*error + Ki*sum*dt +
Kd*(error-olderror)/dt;
olderror = error;
oldsum = sum;
```

A number of elements are missing from this algorithm, including saturation control for the integral term and filtering for the derivative term, but the fundamental concept should be apparent. Note that *dt* is the rate at which we update the PWM generator in the drive. This rate is dependent on the application. You would need a faster servo update rate to servo, say, a feather than you would for a battleship.

Phase lag

Because phase lag is so important to the stability of a servo system, let's take a moment to clarify its meaning. A phase lag might be described as delay between a reference signal and a measured signal. In signal processing, we sometimes call this phase angle, phase delay, or phase displacement.

Periodic signals are often measured in degrees or radians. In Figure 2a, you see two square waves. The top wave is marked in degrees. Now, suppose one of the waves in Figure 2a is the input to a black box and the other wave is the output. The box does nothing to the signal. It doesn't even delay it. These two signals are *in phase*, which means they begin at the same point and both edges are positive.

Figure 2b illustrates a black box with a fixed delay. If we input our

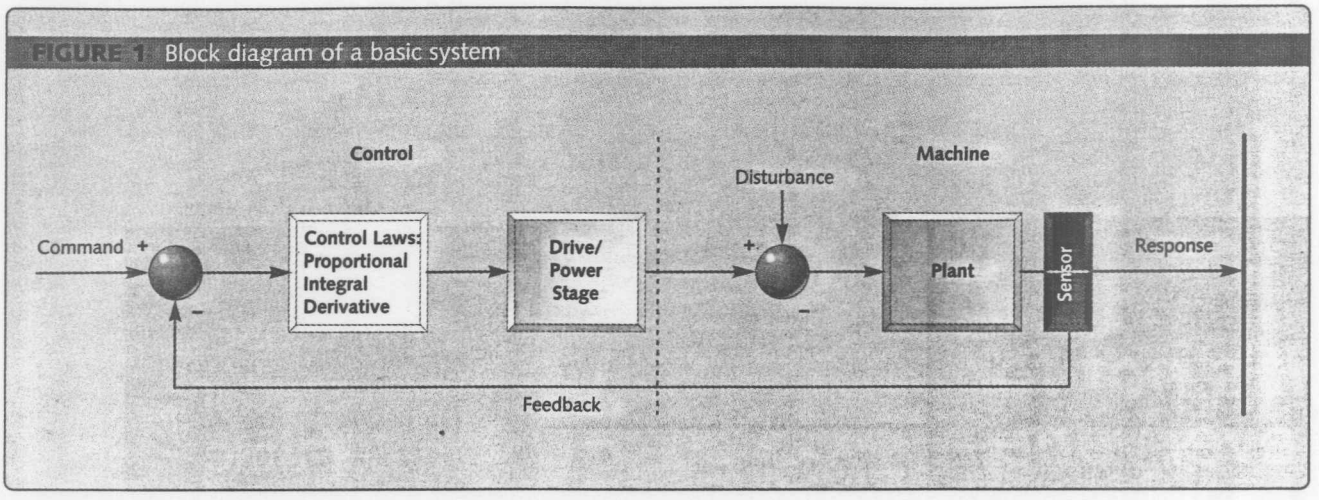


FIGURE 1 Block diagram of a basic system

square wave and examine the output, it will always exhibit a fixed delay. We can quantify the delay in degrees by noticing at what point in the periodic cycle it begins and continues as the reference signal does. Just because a delay is fixed in time-does not mean that phase lag is fixed as well. For example, if our fixed delay is half a second and we input a square wave with a period of two seconds, the phase lag is 90 degrees. But, if we input a square wave with a one second period instead, the phase lag becomes 180 degrees. This is were the trouble arises.

Zero phase lag is impossible in a system such as the one we are describing, because each of the system's elements contribute. The motor itself contributes an L/R time constant; currents and velocity do not change instantaneously with the application of power. The feedback system creates delay. Filters, drivers, and processes such as integration also cause delay. A system's stability and bandwidths depend on the choices you make in each of these areas.

Step response

As with other signal processing activities, control theory is based in the frequency domain and reflected in the time domain. But in control theory, perhaps more than signal processing, time domain responses are important and often used to judge the performance of a system. Here, we'll use the step response to describe a system in practical or industrial applications because it's easy to measure with an oscilloscope.

Typically, the stability of a system is measured by its step response. Step response is a system's response to any instantaneous change at the input. The change might be a command to a motor to move to a new location and stop: a square wave. The characteristic we are most concerned with is overshoot. Overshoot is the ratio of the peak of the response to the command. The amount of acceptable overshoot varies with the application.

System stability is dependent on delays and phase. In order for a control system to work, the feedback must be negative at the summing node you see in Figure 1. If it is not, the system will become forward-driven, the error will be amplified by the gains applied, and the system will go out of control. Anyone who has ever reversed the encoder wiring on a servo knows what this is like.

Building up enough phase lag throughout the system can produce a similar result. When this is the case, the feedback will appear to go positive at a certain frequency—phase lags are frequency dependent—and destabilize the system. (Remember, all it takes for the feedback to appear positive at the summing node is a phase-lag of 180 degrees.) Such a system can be very sensitive to noise, since these are usually high frequency elements, and may cause the application to perform poorly even at low frequencies. As I describe the components of the PID algorithm, I will indicate how they affect the phase in the system.

Proportional control

Proportional control, or P control, is the simplest control technique. It takes the sum of the negative feedback and the command as its input. The error is the difference between the commond and the actual position. The control applies a gain and the result is sourced to the drive:

$$U[t] = K_p * e[t]$$

P control is easy to work with, but it makes no compensation for disturbances such as temperature and friction, among others. These long-term errors are sometimes referred to as DC errors. The phase lag of a proportional gain is zero degrees.

To tune a system such as this, apply the square wave step command and raise the gain until it becomes unsta-

ble. Choose an acceptable gain that results in a stable system.

Integral control

Integral control, or I control, can correct the DC errors of the proportional control. The integral gain provides low-frequency stiffness. The most common form of integration is performed as:

$$I_n = I_{n-1} + dt * e[t]$$

The error is scaled by time and summed with the previous output. The result is multiplied by a gain factor:

$$U[t] = K_i * I_n$$

The higher the integral gain, the faster the control will correct. Keep in mind that the addition of integral gain to proportional gain can result in overshoot and ringing, which reduce bandwidth, especially with higher integral gains.

The inherent 90 degree phase lag, combined with the lag in the plant and feedback systems can cause the system to be unstable, if the gain and frequency are high enough.

PI control

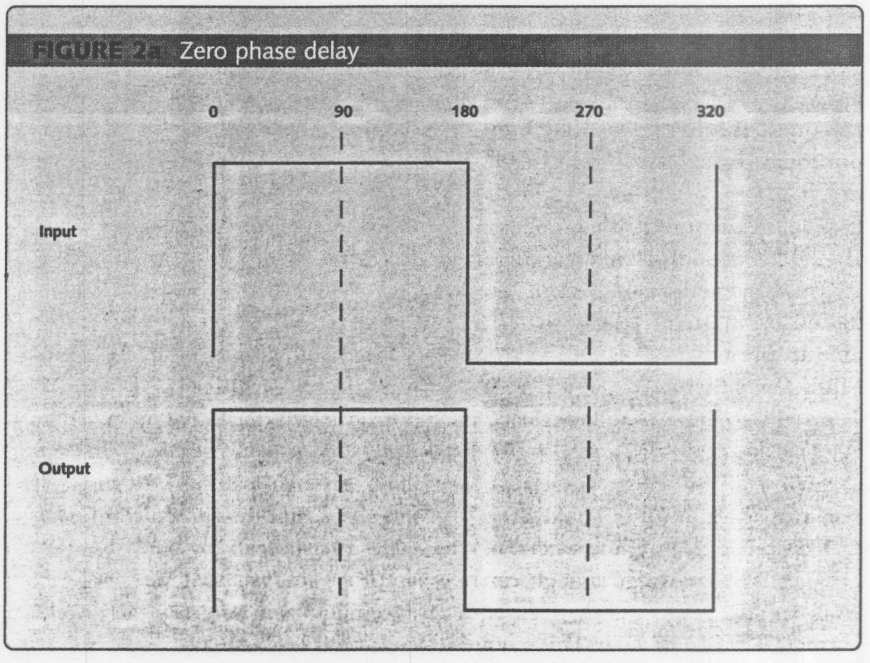
Proportional integral, or PI, control is popular because it provides the response of a simple P control with the ability to eliminate offsets stemming from disturbances. The output of this controller is the sum of two signals, one a scaled proportional and the other a scaled integral:

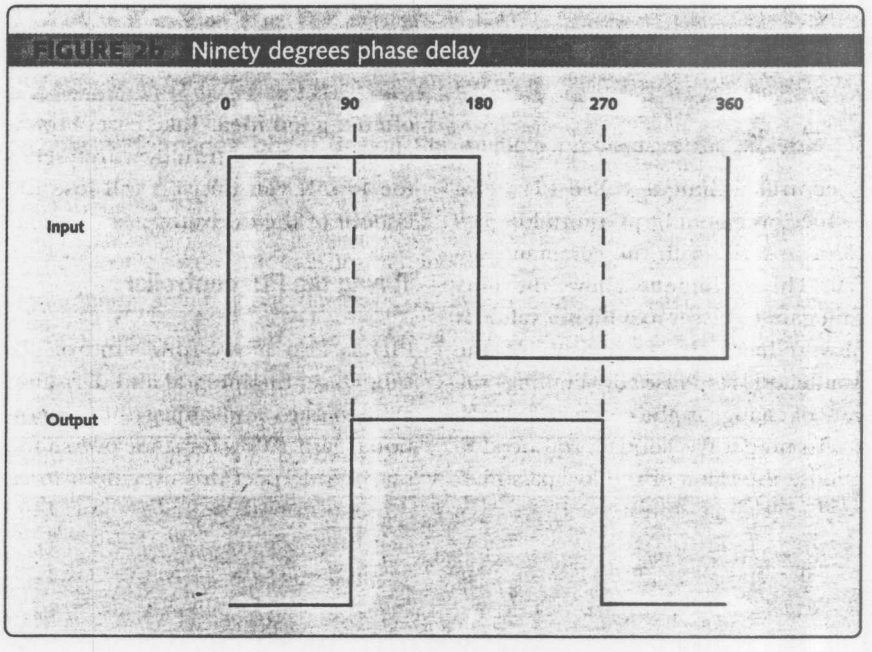
$$U[t] = K_p * e[t] + K_i * I_n$$

Tuning a PI controller

We tune a PI controller in two zones: high and low. We tune the proportional gain in the high zone and the integral gain in the low zone.

Proportional integral, or PI, control is popular because it provides the response of a simple P control with the ability to eliminate offsets stemming from disturbances. The output of this controller is the sum of two signals, one a scaled proportional and the other a scaled integral.





First, tune the high end as you did with the simple P controller described previously. This is done with the integral gain set to zero. Watch for instabilities resulting from accumulated phase lag in the loop.

For the integral tuning, you will again use the square wave command (input); leaving the proportional gain as desired, increase the gain of the integral control until you have reached the amount of overshoot acceptable

To tune a PI+ control, you need to choose the value of the low pass filter on the command input before tuning the integral. After that's done, its tuning is similar to the PI controller.

for your system. This will probably be less than 30%. A PI controller will always generate some overshoot.

You do not have to retune the proportional gain after tuning the integral gain.

Zone-based tuning is a popular and workable technique for separating out the control elements for tuning. For a discussion in greater detail than I am able to present here, consult George Ellis's *Control System Design Guide.*

At this point we will neglect any changes in gain regarding the plant (or motor). These are usually due to temperature or current and should be anticipated in the design of the system. Such changes, if they are large enough, can be compensated for with a technique known as gain scheduling. This requires changing the proportional gain on the fly. Its implementation depends on the controller you are using.

PI+

A control technique called PI+ helps reduce overshoot by putting a low pass filter in series with the command signal. This techinique allows the integral gain to be set to a higher value. It also reduces the bandwidth of the command response by limiting the rate of change in the command signal.

To tune a PI+ control, you need to choose the value of the low pass filter on the command input before tuning the integral. After that's done, its tuning is similar to the PI controller. Obviously, setting the turn-over frequency (3dB point) for this filter as high as possible will provide the greatest response and stiffness to the system.

Differential

The addition of a differential gain can increase system responsiveness significantly, doubling it in some cases. It also introduces a phase lead of 90 degrees. A typical digital implementation is:

$$D_n = \frac{e[n] - e[n-1]}{dt}$$

This formula communicates one of the more significant problems of introducing derivative gain. As you can see, it represents a difference between error samples. At higher frequencies that difference can be large. The phase lead it contributes can improve low frequency response, but too high a gain can result in instability at high frequencies.

Noise can also cause a problem for derivative terms, because the derivative amplifies it. For that reason, a filter following the derivative is often a good idea. Take care, however, not to set the turn-over frequency too low. If you do, you will lose the benefit of the derivative.

Tuning the PID controller

PID is also a two-zone control. To tune it, set the integral and derivative gain to zero and adjust the proportional gain. Allow for some overshoot, say 10%. Expect this overshoot to be handled when you introduce the derivative gain.

Next, raise the derivative gain until the overshoot is gone. Finally, increase the integral gain until you have an overshoot acceptable to your system (maybe 10% to 15%).

Problems here will probably come from noise. If the noise cannot be eliminated at the source, consider raising the resolution of the feedback sensor. If this cannot be done, you will probably need to reduce the proportional or derivative gains.

PID+

A PID+ controller is basically the PID controller from the last section with the addition of a low pass filter on the command input. The tuning is the same as for the PID controller, except that you need to choose the value of the low pass filter for the command input before tuning the integral gain.

PD

A PD controller is the PID with the integral gain set to zero. Tune as with the PID controller; the problems will be the same.

I hope these tips provide some help in this complex area. Of course, PID is not the only way to control motion, as PWM is not the only way to source power to the plant. These topics hold plenty of material for future columns.

Next month, we'll examine some ways to use a digital signal processor to perform the functions of resolvers and sinusoidal converters for high-resolution encoders. **esp**

Don Morgan is senior engineer at Ultra Stereo Labs and a consultant with 25 years experience in signal processing, embedded systems, hardware, and software. Morgan recently completed a book about numerical methods, featuring multi-rate signal processing and wavelets, called Numerical Methods for DSP Systems in C. *He is also the author of* Practical DSP Modeling, Techniques, and Programming in C, *published by John Wiley & Sons, and* Numerical Methods for Embedded Systems *from M&T. Don's e-mail address is dgm@baykitty.com.*

Resources

Ellis, George. *Control System Design Guide*. London: Academic Press, 2000.